

(19)



JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: **06202875 A**

(43) Date of publication of application: **22.07.94**

(51) Int. Cl

**G06F 9/45**

(21) Application number: **04360619**

(71) Applicant: **NEC CORP**

(22) Date of filing: **28.12.92**

(72) Inventor: **YAMAMOTO KUNIKO**

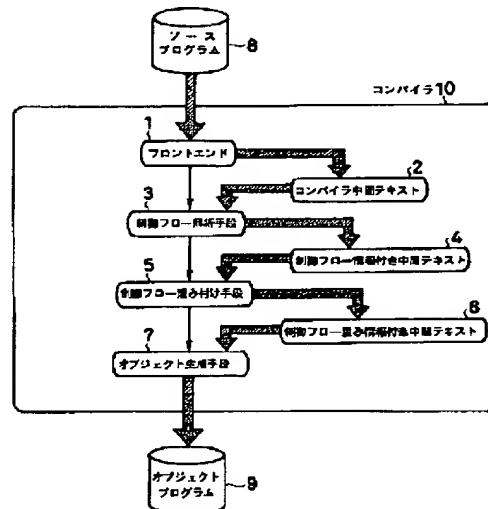
**(54) COMPILER FOR ATTAINING OPTIMIZATION  
WITH IN-LINE EXPANSION**

**(57) Abstract:**

**PURPOSE:** To decide the necessity of in-line expansion for each function call, to perform in-line expansion corresponding to an execution image and to enable practical and efficient in-line expansion.

**CONSTITUTION:** A control flow weighting means 5 estimates the number of times of executing each partial control flow constituting a control flow based on the analyzed result of a control flow analyzing means 3 and weights each partial control flow based on the number of times of execution. An object generating means 7 decides the necessity of in-line expansion for a function called by each function call while referring to the weighted result of the control flow weighting means 5 and generates an object program 9 reflected with the decision.

**COPYRIGHT:** (C)1994,JPO&Japio



(19)日本国特許庁 (JP)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開平6-202875

(43)公開日 平成6年(1994)7月22日

(51)Int.Cl.<sup>5</sup>

G 0 6 F 9/45

識別記号

府内整理番号

F I

技術表示箇所

9292-5B

G 0 6 F 9/ 44

3 2 2 F

審査請求 有 請求項の数5 (全8頁)

(21)出願番号 特願平4-360619

(22)出願日 平成4年(1992)12月28日

(71)出願人 000004237

日本電気株式会社

東京都港区芝五丁目7番1号

(72)発明者 山本 久仁子

東京都港区芝五丁目7番1号 日本電気株式会社内

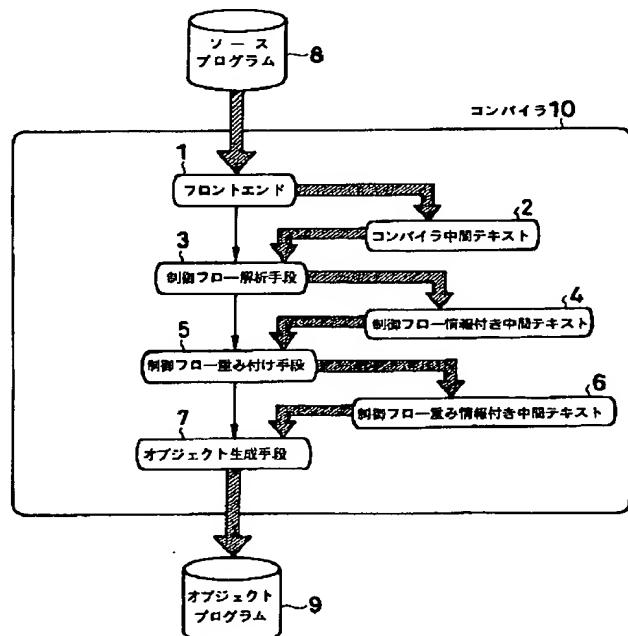
(74)代理人 弁理士 河原 純一

(54)【発明の名称】 インライン展開による最適化を行うコンバイラ

(57)【要約】

【目的】 関数呼出し毎のインライン展開の要否の決定を可能とし、実行イメージに合ったインライン展開を行い、実用的かつ効率的なインライン展開を可能とする。

【構成】 制御フロー重み付け手段5は、制御フロー解析手段3の解析結果に基づいて制御フローを構成する各部分制御フローの実行回数を推測し、その実行回数に基づいて各部分制御フローに対する重み付けを行う。オブジェクト生成手段7は、制御フロー重み付け手段5による重み付けの結果を参照して各関数呼出しによって呼び出される関数のインライン展開の要否を決定し、その決定を反映したオブジェクトプログラム9を生成する。



## 【特許請求の範囲】

【請求項1】 制御フロー解析手段の解析結果に基づいて制御フローを構成する各部分制御フローの実行回数を推測し、その実行回数に基づいて各部分制御フローに対する重み付けを行う制御フロー重み付け手段と、この制御フロー重み付け手段による重み付けの結果を参照して各関数呼出しによって呼び出される関数のオンライン展開の要否を決定し、その決定を反映したオブジェクトプログラムを生成するオブジェクト生成手段とを有することを特徴とするオンライン展開による最適化を行うコンパイラ。

【請求項2】 重み付けの対象の部分制御フローに関するループについてのループ回数およびネスト数を使って当該部分制御フローの実行回数を推測し、ループをキーとする重み付けを当該部分制御フローに対して行う前記制御フロー重み付け手段を有することを特徴とする請求項1記載のオンライン展開による最適化を行うコンパイラ。

【請求項3】 処理対象の関数呼出しを含む部分制御フローに対する前記制御フロー重み付け手段による重み付けの結果と当該関数呼出しによって呼び出される関数のオブジェクトサイズとを使った算出式を用いてオンライン展開判定値を算出し、そのオンライン展開判定値とオンライン展開要否指標値との比較に基づいて当該関数呼出しによって呼び出される関数のオンライン展開の要否を決定する前記オブジェクト生成手段を有することを特徴とする請求項1記載のオンライン展開による最適化を行うコンパイラ。

【請求項4】 重み付けの対象の部分制御フローに関するループについてのループ回数が静的に決まらない場合に使用されるデフォルトループ回数をオプション情報に基づいて取得する前記制御フロー重み付け手段を有することを特徴とする請求項2記載のオンライン展開による最適化を行うコンパイラ。

【請求項5】 インライン展開要否指標値をオプション情報に基づいて取得する前記オブジェクト生成手段を有することを特徴とする請求項3記載のオンライン展開による最適化を行うコンパイラ。

## 【発明の詳細な説明】

## 【0001】

【産業上の利用分野】 本発明は、コンパイラに関し、特に最適化の一手法である「関数のオンライン展開」を行うコンパイラ（オンライン展開による最適化を行うコンパイラ）に関する。

## 【0002】

【従来の技術】 従来、オンライン展開による最適化を行うコンパイラは、以下のようにしてオンライン展開を行うべきか否か（オンライン展開の要否）を決定していた。

## 【0003】 一の手法では、オンライン展開の要否

の指示が他の種類の最適化の要否の指示とともに最適化オプション（オブジェクトプログラムの実行効率をどこまで向上させるかに基づく最適化オプション）によって指定され、その最適化オプションに基づいてオンライン展開の要否が決定される。この最適化オプションの指定は、プログラム全体に対して行われる（関数呼出し毎に行われることはない）。最適化オプションには、最適化レベルを指定することができるものが多い。

【0004】 他の手法では、コンパイル時にオンライン展開のみについての要否の指示が所定のオプションによって指定され、そのオプションに基づいてオンライン展開の要否が決定される。この場合にも、このオプションの指定は、プログラム全体に対して行われ、関数呼出し毎に行われることはない。

【0005】 なお、プログラム全体に対してオンライン展開を行うべきことが指定された場合には、各関数のオンライン展開を行うべきか否かがオブジェクトプログラムのレベルでの当該関数の大きさ（当該関数のオブジェクトサイズ）によって決定されていた。

【0006】 上述のように、従来の技術では、オンライン展開の要否の決定は、関数呼出し毎ではなくプログラム全体を対象として行われており（オンライン展開が画一的に適用されており）、関数のオブジェクトサイズという要因しか考慮されていなかった（関数の呼出し回数等の他の要因は全く考慮されていなかった）。

【0007】 また、上述した従来の技術に対する新技術として公開されているものには、以下に示す技術がある。以下、上述した従来の技術を「旧従来技術」といい、以下に示す技術を「新従来技術」という。

【0008】 「特開平3-99330（手続きオンライン展開方式）」に係る特許出願によって示される新従来技術：中間テキストのレベルでカウントされた関数の静的な参照回数（中間テキストに出現する当該関数の参照回数）を使用して、その関数のオンライン展開の要否を決定する。

【0009】 「特開平1-118931（プログラム変換方式）」に係る特許出願によって示される新従来技術：オンライン展開による最適化を行わずに仮の実行プログラムを生成して実行し、その実行プログラムの実行によって得られる動的な（実際の）関数の呼出し回数やループの実行回数（ループ回数）を使用してオンライン展開の要否を決定する。

## 【0010】

【発明が解決しようとする課題】 上述したように、旧従来技術では、オンライン展開の要否の決定がプログラム全体を対象として行われており、関数のオブジェクトサイズ以外の要因（関数の呼出し回数等の要因）が考慮されていない。したがって、オブジェクトプログラムのサイズの増大等が生じてオンライン展開が真の意味での最適化に寄与しない可能性がある（オンライン展開によつ

てオブジェクトプログラムの実行効率を逆に落とすことがある)という欠点があった。

【0011】また、「特開平3-99330」に係る特許出願によって示される新従来技術では、オンライン展開の要否を決定する際に新たに関数の参照回数という要因が考慮されているが、この参照回数は中間テキストに出現する参照回数であり実際の関数の呼出し回数とは異なるものである。したがって、ある関数に対する関数呼出しが中間テキスト中に何度も現れれば実際には一度もその関数が呼び出されなくてもオンライン展開が行われてしまい(実行イメージに合わないオンライン展開による最適化が行われることとなり)、無用にオブジェクトプログラムのサイズが大きくなつてオブジェクトプログラムの実行効率の低下を招く可能性があるという欠点があった。

【0012】さらに、「特開平1-118931」に係る特許出願によって示される新従来技術では、仮の実行プログラムを実際に実行した結果を使用してオンライン展開の要否が決定されており、「仮の実行プログラムの生成および実行」という大掛かりな処理(一般的なコンパイラが持つ通常の最適化処理の範囲を逸脱する処理)が必要になる。したがって、実用的かつ効率的なオンライン展開による最適化を行うことができないという欠点があった。

【0013】本発明の目的は、上述の点に鑑み、関数呼出し毎にオンライン展開の要否の決定を行うことを可能とし、実行イメージに合つたオンライン展開を行うことができ、実用的かつ効率的なオンライン展開を可能とする「オンライン展開による最適化を行うコンパイラ」を提供することにある。

#### 【0014】

【課題を解決するための手段】本発明のオンライン展開による最適化を行うコンパイラは、制御フロー解析手段の解析結果に基づいて制御フローを構成する各部分制御フローの実行回数を推測しその実行回数に基づいて各部分制御フローに対する重み付けを行う制御フロー重み付け手段と、この制御フロー重み付け手段による重み付けの結果を参照して各関数呼出しによって呼び出される関数のオンライン展開の要否を決定しその決定を反映したオブジェクトプログラムを生成するオブジェクト生成手段とを有する。

#### 【0015】

【実施例】次に、本発明について図面を参照して詳細に説明する。

【0016】図1は、本発明のオンライン展開による最適化を行うコンパイラの第1の実施例(コンパイラ10)の構成等を示すブロック図である。

【0017】コンパイラ10は、フロントエンド(字句解析部、構文解析部および意味解析部等の総称)1と、コンパイラ中間テキスト2と、制御フロー解析手段3

と、制御フロー情報付き中間テキスト4と、制御フロー重み付け手段5と、制御フロー重み情報付き中間テキスト6と、オブジェクト生成手段7とを含んで構成されている。また、コンパイラ10は、ソースプログラム8を入力して、オブジェクトプログラム9を出力する。

【0018】図2は、制御フロー重み付け手段5の詳細な構成等を示すブロック図である。制御フロー重み付け手段5は、制御フロー情報付き中間テキスト4を入力するループ検出部51と、ループ回数判定部52と、制御フロー重み情報付き中間テキスト6を出力する制御フロー重み付け部53とを含んで構成されている。

【0019】図3は、オブジェクト生成手段7の詳細な構成等を示すブロック図である。オブジェクト生成手段7は、関数オブジェクトサイズSを出力する呼出し関数サイズ算出部71と、制御フロー重み情報付き中間テキストによって示される部分制御フローの重み値Nと関数オブジェクトサイズSとを入力してオンライン展開判定値Jを出力するオンライン展開判定値算出部72と、オンライン展開判定値Jを入力する関数オンライン展開部73とを含んで構成されている。なお、オブジェクト生成手段7は、オンライン展開処理に関する機能(図3中の破線の枠内の機能)以外にもオブジェクトプログラム9の生成に関する多くの機能を有している。

【0020】次に、このように構成された本実施例のオンライン展開による最適化を行うコンパイラ(コンパイラ10)の動作について説明する。

【0021】フロントエンド1は、ソースプログラム8を入力し、従来技術と同様の方法で処理(字句解析、構文解析および意味解析等)を行い、コンパイラ中間テキスト2を出力する。

【0022】制御フロー解析手段3は、コンパイラ中間テキスト2を入力し、従来技術と同様の方法で制御フロー解析を行い、制御フロー情報付き中間テキスト4(制御フロー解析の結果である制御フロー情報を有する中間テキスト)を出力する。

【0023】制御フロー重み付け手段5内の各部は、以下に示すような処理を行う(図2参照)。

【0024】ループ検出部51は、制御フロー情報付き中間テキスト4を入力し、制御フロー情報によって示される制御フローからループ(ループ構造)を検出する。

【0025】ループ回数判定部52は、ループ検出部51によって検出されたループの各々について、次のようにしてループ回数(ループの実行回数)を求める。

【0026】処理対象のループを構成する条件分岐テキストを探し、その条件分岐テキストにおける条件が「変数V 比較演算 定数C1」という形であるか否かを判定する(この判定は、「静的にループ回数を求めることができるか否か」の判定となる)。

【0027】当該条件が「変数V 比較演算 定数C1」という形である場合には、処理対象のループの入

口から制御フローをさかのぼり、変数Vに値を設定しているテキストを探す。

【0028】 そのテキストにおいて、変数Vに設定される値が定数C2であり、ループを制御するインダクション変数の増減値が定数C3である場合に、それらの定数C2およびC3と当該条件中の定数C1とを比較し、その比較に基づいてループ回数L（定数值）を求める。

【0029】 以上のようにして静的にループ回数Lを求めることができない場合（の判定で当該条件が「変数V 比較演算 定数C1」という形でない場合、において変数Vに設定される値が定数でない場合、またはにおいてループを制御するインダクション変数の増減値が定数でない場合）には、コンパイラ10におけるループ回数のデフォルト値であるデフォルトループ回数 $\alpha$ をループ回数Lとする。

【0030】 制御フロー重み付け部53は、ループ回数判定部52によって求められたループ回数Lを用いて、以下に示すようにして、制御フローを構成する各部分制御フローに対する重み付け（重み値Nの設定）を行う。なお、部分制御フローとは、最初のテキストから最後のテキストまでに分岐を示すテキストおよびラベルを示すテキストが存在せず連続して実行される制御フロー中の部分をいう。

【0031】 処理対象の部分制御フローがループに含まれない場合（ループの外側に存在する場合）には、その部分制御フローの重み値Nを1とする。

【0032】 処理対象の部分制御フローがネスト数n（nは正整数）のループ群に含まれる場合には、その部分制御フローの重み値Nを $\prod L_i$ （ $L_1 \times L_2 \times \dots \times L_n$ 。 $i = 1, 2, \dots, n$ であり $L_i$ はi層目のネストのループのループ回数）とする。

【0033】 すなわち、制御フロー重み付け部53は、部分制御フローに関するループ（部分制御フローを含むループ）についてのループ回数およびネスト数（部分制御フローがループに含まれるか否かという情報を含む）を使って当該部分制御フローの実行回数を推測し、ループをキーとする重み付け（重み値Nの設定）を当該部分制御フローに対して行う。

【0034】 制御フロー重み付け部53は、上述のようにして設定した重み値Nを示す情報を有する中間テキストである制御フロー重み情報付き中間テキスト6を出力する。

【0035】 オブジェクト生成手段7は、従来通り、レジスタ割付け、コード生成および最適化等の処理を行う。オブジェクト生成手段7内の図3に示す各部は、このような処理の過程で、最適化処理の中の1つであるインライン展開処理を行う。このインライン展開処理は、最適化処理の過程で関数呼出しが検出された場合に起動され、以下に示すようにして行われる。

【0036】 呼出し関数サイズ算出部71は、処理対象の関数呼出しによって呼び出される関数を探し、その関数のオブジェクトサイズSを算出する。

【0037】 インライン展開判定値算出部72は、当該関数呼出しを含む部分制御フローの重み値N（制御フロー重み付け手段5によって求められた値）と当該関数呼出しによって呼び出される関数のオブジェクトサイズSとにに基づいて、インライン展開判定値（インライン展開の要否を判定するための値）Jを算出する。NとSとともにJを導く算出式は、ソースプログラム8の対象言語（ソースプログラム8を記述するプログラム言語）の特徴やオブジェクトプログラム9が実行されるターゲットマシンの特徴等に基づいて決定される。

【0038】 関数インライン展開部73は、インライン展開判定値Jとコンパイラ10におけるインライン展開要否判定用のデフォルト値であるインライン展開要否指標値 $\beta$ とを比較する。その比較で「 $J > \beta$ 」ならば、処理対象の関数呼出しを「インライン展開の対象とした方がよい関数呼出し」とみなして、その関数呼出しを関数

（その関数呼出しによって呼び出される関数）の本体のオブジェクトテキストで置換する。

【0039】 オブジェクト生成手段7は、以上のようなインライン展開処理を行った上で、オブジェクトプログラム9を出力する。

【0040】 なお、本実施例に係るコンパイラ10では、制御フロー重み付け手段5が請求項2記載の発明における制御フロー重み付け手段で実現される場合について述べた。しかし、制御フロー解析手段3の解析結果に基づいて何らかの方法で部分制御フローの実行回数を推測して各部分制御フローに対する重み付けを行うものであれば、請求項1記載の発明における制御フロー重み付け手段5に限られるものではない。

【0041】 また、本実施例に係るコンパイラ10では、オブジェクト生成手段7が請求項3記載の発明におけるオブジェクト生成手段で実現される場合について述べた。しかし、重み付けの結果を参照して各関数呼出しによって呼び出される関数のインライン展開の要否を決定するものであれば、請求項1記載の発明におけるオブジェクト生成手段は本実施例におけるオブジェクト生成手段7に限られるものではない。

【0042】 図4は、本発明のインライン展開による最適化を行うコンパイラの第2の実施例（コンパイラ20）の構成等を示すブロック図である。

【0043】 コンパイラ20は、フロントエンド1と、コンパイラ中間テキスト2と、制御フロー解析手段3と、制御フロー情報付き中間テキスト4と、制御フロー重み付け手段5と、制御フロー重み情報付き中間テキスト6と、オブジェクト生成手段7と、オプション情報1とを含んで構成されている。また、コンパイラ20

は、ソースプログラム8を入力して、オブジェクトプログラム9を出力する。すなわち、本実施例（第2の実施例）に係るコンパイラ20は、図1に示す第1の実施例に係るコンパイラ10に対してオプション情報11が追加された構成になっている。したがって、オプション情報11に関する処理以外は、本実施例における処理と第1の実施例における処理とは同様である。この意味で、図1と図4とで対応する構成要素（コンパイラ10および20自身は除く）については同一の符号を用いて示している。

【0044】なお、図2および図3は、本実施例の制御フロー重み付け手段5およびオブジェクト生成手段7の詳細な構成等を示すブロック図である。

【0045】次に、このように構成された本実施例のインライン展開による最適化を行うコンパイラ（コンパイラ20）の動作について説明する。基本的な動作の流れはコンパイラ10の動作の流れと同様であるので、ここではコンパイラ10の動作と異なる動作についてのみ列挙する。

【0046】フロントエンド1は、ソースプログラム8を入力してコンパイラ中間テキスト2を出力する際に、コンパイラ20の起動時にユーザによって与えられたオプション（指定ループ回数および指定指標値）を有するオプション情報11を生成する。

【0047】制御フロー重み付け手段5内のループ回数判定部52は、静的にループ回数を決めることができない場合に、オプション情報11に基づいてデフォルトループ回数 $\alpha$ を求める。すなわち、ユーザによって指定された指定ループ回数をデフォルトループ回数 $\alpha$ として使用する。

【0048】オブジェクト生成手段7内の関数インライン展開部73は、オプション情報11に基づいてインライン展開要否指標値 $\beta$ を取得して、そのインライン展開要否指標値 $\beta$ とインライン展開判定値Jとの比較を行う。すなわち、ユーザによって指定された指定指標値をインライン展開要否指標値 $\beta$ として使用する。

【0049】このように、本実施例では、ユーザはコンパイル時にソースプログラム8の特性やユーザの要求等に応じてデフォルトループ回数やインライン展開要否指標値を指定することができる。

【0050】なお、ユーザによってオプションとして与えられる情報は、本実施例のように指定ループ回数および指定指標値の両方である場合だけではなく、指定ループ回数および指定指標値のいずれか一方だけである場合も許容される。

【0051】

\* 【発明の効果】以上説明したように本発明は、旧従来技術においてプログラム全体を対象として行われていたインライン展開の要否の決定を関数呼出し毎に行うことにより、より詳細な情報によって真の意味での最適化（インライン展開による最適化）の実現の可能性が高くなるという効果がある。

【0052】また、関数呼出し毎にインライン展開を行うか否かを決定する要因として制御フロー解析に基づいて求められる部分制御フローの実行回数を使っているの

10 で、実際の実行イメージに合わない最適化（「特開平3-99330」に係る特許出願によって示される中間テキスト上の関数の参照回数を使う最適化等）や通常のコンパイラの最適化処理の範囲を逸脱する処理が必要になる最適化（「特開平1-118931」に係る特許出願によって示される大掛かりな仕組みを用いる最適化等）を避けることができ、実行イメージに適合するとともに実用的かつ効率的なものとなる最適化（インライン展開による最適化）を行うことができるという効果がある。

【0053】さらに、インライン展開の要否のキーとなる値をユーザがオプションによって指定することを可能とすることにより（請求項4および請求項5参照）、ソースプログラムの特性やユーザの要求等に応じたインライン展開による最適化を行うことができるようになるという効果がある。

#### 【図面の簡単な説明】

【図1】本発明の第1の実施例の構成等を示すブロック図である。

【図2】図1および図4中の制御フロー重み付け手段の詳細な構成等を示すブロック図である。

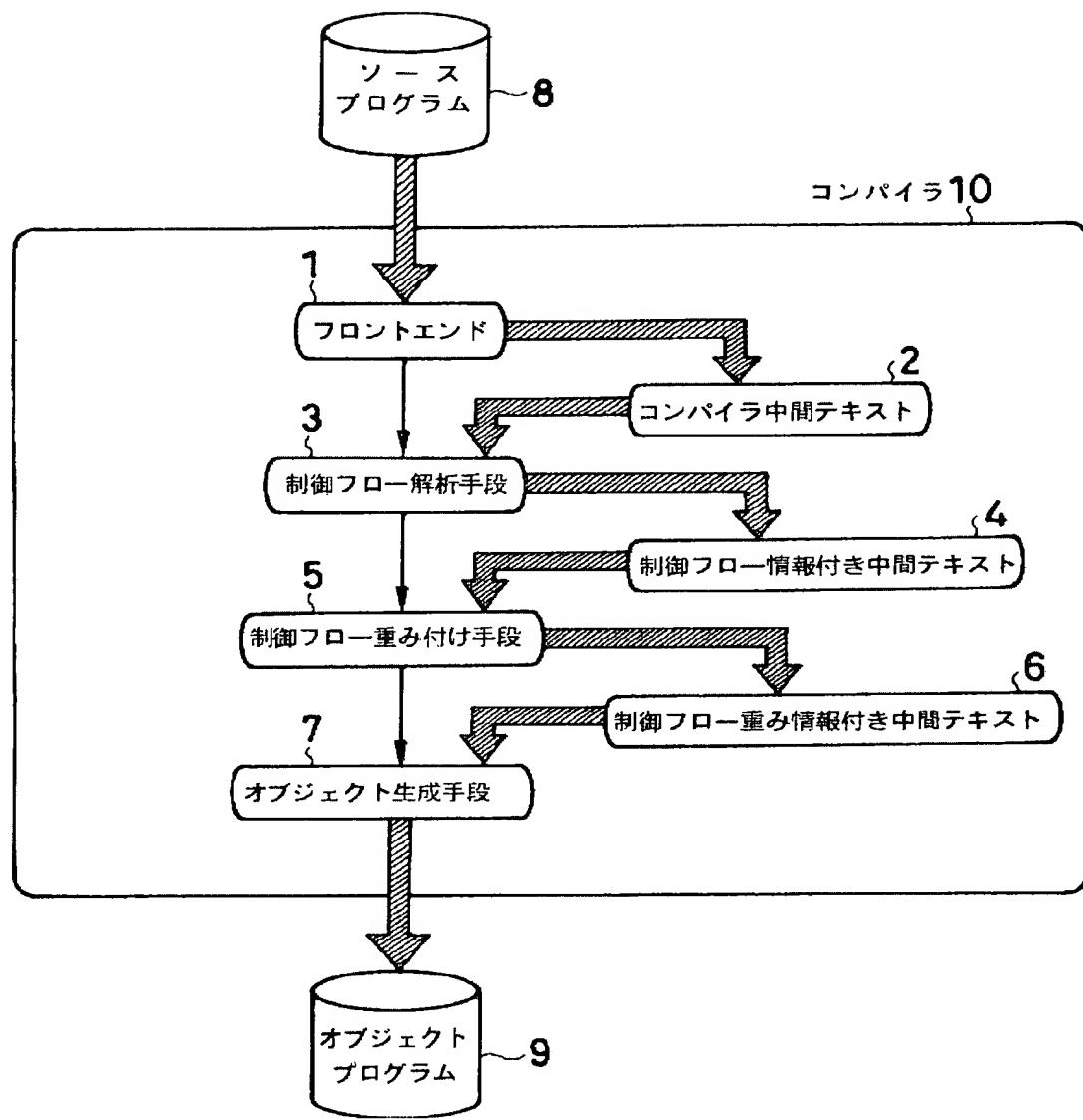
30 【図3】図1および図4中のオブジェクト生成手段の詳細な構成等を示すブロック図である。

【図4】本発明の第2の実施例の構成等を示すブロック図である。

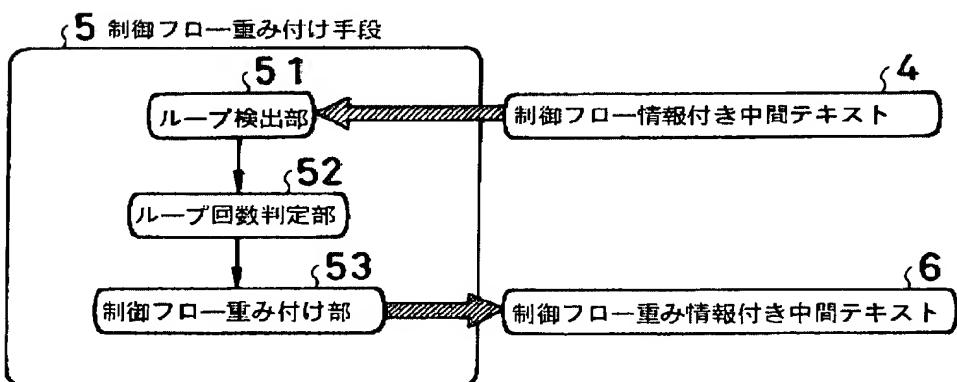
#### 【符号の説明】

- 1 フロントエンド
- 2 コンパイラ中間テキスト
- 3 制御フロー解析手段
- 4 制御フロー情報付き中間テキスト
- 5 制御フロー重み付け手段
- 40 6 制御フロー重み情報付き中間テキスト
- 7 オブジェクト生成手段
- 8 ソースプログラム
- 9 オブジェクトプログラム
- 10, 20 コンパイラ
- 11 オプション情報

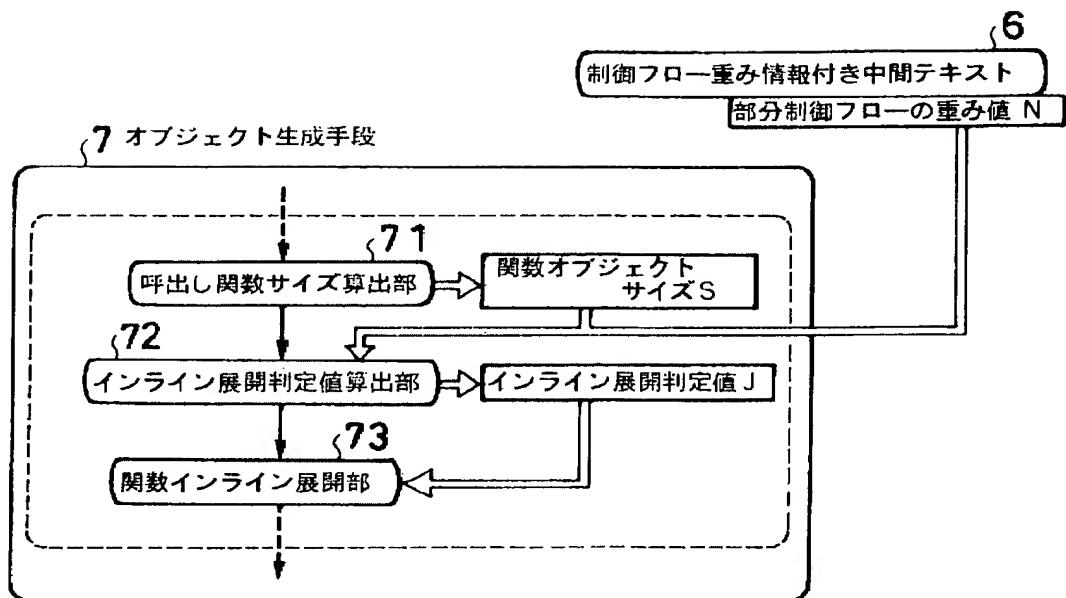
【図1】



【図2】



【図3】



【図4】

